PIPELINE COPROCESSOR

Publication number: KR20050084628 (A)
Publication date: 2005-08-26
Inventor(s): MATHUR CHAN DAN [US]; HELLENBACH SCOTT [US]; RAPP JOHN W [US]; JACKSON LARRY [US]; JONES MARK [US]; CHERASARO TROY [US]
Applicant(s): LOCKHEED CORP [US]
Classification:
- International: ***G06F9/30; G06F9/38; G06F9/445; G06F9/46; G06F15/78;*** **G06F9/30; G06F9/38; G06F9/445; G06F9/46; G06F15/76;** (IPC1-7): G06F9/30; G06F9/38
- European: G06F9/38S1
Application number: KR20057007748 20050430
Priority number(s): US20030683929 20031009; US20030683932 20031009; US20030684053 20031009; US20030684057 20031009; US20030684102 20031009; US20020422503P 20021031

Also published as:
WO2004042560 (A2)
WO2004042560 (A3)
WO2004042574 (A2)
WO2004042574 (A3)
WO2004042569 (A2)
more >>

Abstract not available for KR 20050084628 (A)
Abstract of corresponding document: **WO 2004042560 (A2)**

A peer-vector machine includes a host processor and a hardwired pipeline accelerator. The host processor executes a program, and, in response to the program, generates host data, and the pipeline accelerator generates pipeline data from the host data. Alternatively, the pipeline accelerator generates the pipeline data, and the host processor generates the host data from the pipeline data. Because the peer-vector machine includes both a processor and a pipeline accelerator, it can often process data more efficiently than a machine that includes only processors or only accelerators. For example, one can design the peer-vector machine so that the host processor performs decision-making and non-mathematically intensive operations and the accelerator performs non-decision-making and mathematically intensive operations ; By shifting the mathematically intensive operations to the accelerator, the peer-vector machine often can, for a given clock frequency, process data at a speed that surpasses the speed at which a processor-only machine can process the data.

(19) KOREAN INTELLECTUAL PROPERTY OFFICE

KOREAN PATENT ABSTRACTS

(11) Publication number: 1020050084628 A

(43) Date of publication of application: 26.08.2005

(21) Application number: 1020057007748

(22) Date of filing: 30.04.2005

(30) Priority: 2003 683929 US 09.10.2003

(51) Int. Cl: *G06F 9/30 (2006.01);*
G06F 9/38 (2006.01);

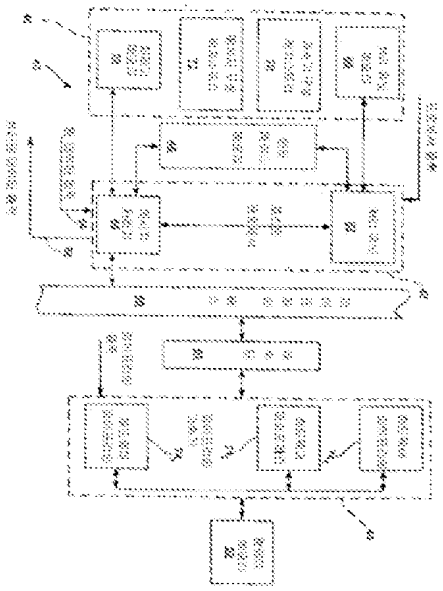
(71) Applicant: LOCKHEED MARTIN CORPORATION

(72) Inventor: MATHUR CHAN DAN
HELLENBACH SCOTT
RAPP JOHN W.
JACKSON LARRY
JONES MARK
CHERASARO TROY

(54) **PIPELINE COPROCESSOR**

(57) Abstract:

A peer-vector machine includes a host processor and a hardwired pipeline accelerator. The host processor executes a program, and, in response to the program, generates host data, and the pipeline accelerator generates pipeline data from the host data. Alternatively, the pipeline accelerator generates the pipeline data, and the host processor generates the host data from the pipeline data. Because the peer-vector machine includes both a processor and a pipeline accelerator, it can often process data more efficiently than a machine that includes only processors or only accelerators. For example, one can design the peer-vector machine so that the host processor performs decision-making and non-mathematically intensive operations and the accelerator performs non-decision-making and mathematically intensive operations. By shifting the mathematically intensive operations to the accelerator, the peer-vector machine often can, for a given clock frequency, process data at a speed that surpasses the speed at which a processor-only machine can process the data.



This Facsimile First Page has been artificially created from the Korean Patent Abstracts CD Rom

(19)대한민국특허청(KR)
(12) 공개특허공보(A)

(51) 。 Int. Cl.[7]
G06F 9/30
G06F 9/38

(11) 공개번호 10-2005-0084628
(43) 공개일자 2005년08월26일

(21) 출원번호 10-2005-7007748
(22) 출원일자 2005년04월30일
번역문 제출일자 2005년04월30일
(86) 국제출원번호 PCT/US2003/034557
국제출원일자 2003년10월31일
(87) 국제공개번호 WO 2004/042560
국제공개일자 2004년05월21일

(30) 우선권주장
10/683,939 2003년10월09일 미국(US)
10/683,932 2003년10월09일 미국(US)
10/684,053 2003년10월09일 미국(US)
10/684,057 2003년10월09일 미국(US)
10/684,102 2003년10월09일 미국(US)
60/422,503 2002년10월31일 미국(US)

(71) 출원인
록히드 마틴 코포레이션
미국 버지니아주 20110, 매나사스, 윤원 드라이브 9500, 메일 드롭 043, 빌딩 400

(72) 발명자
예류, 첸단
미국 버지니아 20109, 매나사스, 프라이베이츠 코트 11162
웰렌바흐, 스콧
미국 버지니아 20106, 아메스빌, 루아일 리지 드라이브 15381
웹, 존 더블유
미국 버지니아 20110, 매나사스, 리버 크레스트 로드 9350
잭슨, 데릭
미국 버지니아 20112, 매나사스 크레스트북 드라이브 13003
존, 마크
미국 버지니아 20120, 셜드레일, 오크메어 플레이스 15342
제라시오, 프로이
미국 버지니아 22701, 컬페퍼, 캐스트발 코트 1524

(74) 대리인
전영일
김주석

심사청구 : 없음

(54) 파이프라인 코프로세서

요약

피어-엑티 머신에는 호스트 프로세서 및 하드와이어드 파이프라인 가속기가 포함되어 있다. 상기 호스트 프로세서는 프로그램을 실행하고, 상기 프로그램에 응답하여, 호스트 데이터를 생성하며, 상기 파이프라인 가속기는 상기 호스트 데이터로부터 파이프라인 데이터를 생성한다. 선택적으로, 상기 파이프라인 가속기는 상기 파이프라인 데이터를 생성하고, 상기

호스트 프로세서가 상기 파이프라인 데이터로부터 상기 호스트 데이터를 생성한다. 상기 피어-벡터 머신에는 프로세서와 파이프라인 가속기가 모두 포함되어 있기 때문에, 프로세서만 포함하거나 또는 가속기만 포함하는 머신보다 훨씬 효과적으로 데이터를 처리할 수 있다. 예를 들어, 설계자는 호스트 프로세서가 결정-형성 및 수학적으로 집중적인 연산을 수행하고 상기 가속기가 비-결정-형성 및 수학적으로 집중적인 연산을 수행하도록고 피어-벡터 머신을 설계할 수 있다. 상기 수학적으로 집중적인 연산을 가속기로 시프트 함으로써, 피어-벡터 머신은 주어진 클록 주파수에서 프로세서만 있는 머신이 데이터를 처리하는 속도보다 훨씬 빠른 속도로 데이터를 처리할 수 있다.

대표도

도 3

색인어

피어-벡터 머신, 파이프라인, 파이프라인 코프로세서

명세서

기술분야

(우선권 주장)

본 출원은 참고문헌으로서 통합되는 2002년 10월 31일 출원된 미국 가출원 제60/422,503호의 우선권을 주장한다.

(관련된 출원과의 상호참조)

본 출원은 발명의 명칭이 "향상된 컴퓨팅 아키텍처를 가지는 컴퓨팅 머신 및 관련 시스템 및 방법" 인 미국 특허출원 제10/684,053호, 발명의 명칭이 "향상된 컴퓨팅 아키텍처를 위한 파이프라인 가속기 및 관련 시스템 및 방법" 인 미국 특허출원 제10/683,929호, 발명의 명칭이 "프로그램가능한 회로 및 관련 컴퓨팅 머신 및 방법" 인 미국 특허출원 제10/684,057호, 및 발명의 명칭이 "다중 파이프라인 유닛을 가지는 파이프라인 가속기 및 관련 컴퓨팅 머신 및 방법" 인 미국 특허출원 제 10/684,932호와 관련이 되어 있으며, 상기 미국 특허출원 발명은 모두 2003년 10월 9일 동일한 권리자에 의해 출원되었으며, 본 명세서에 참고문헌으로 통합된다.

배경기술

상대적으로 짧은 시간에서 상대적으로 대용량의 데이터를 처리하기 위한 일반적인 컴퓨팅 아키텍처(computing architecture)에는 부하를 분배하는 다중 상호접속 프로세서(multiple interconnected processor)가 포함되어 있다. 처리 부하를 분배함으로써, 이들 다중 프로세서들은 주어진 클록 주파수에서 하나의 프로세서가 할 수 있는 것 보다 더욱 빨리 데이터를 처리할 수 있다. 예를 들어, 프로세서 각각은 데이터의 일부를 처리 하거나 또는 처리되는 알고리즘 일부를 실행할 수 있다.

도 1은 다중-프로세서 아키텍처를 갖는 종래의 컴퓨팅 머신(computing machine)(10)의 개략적인 블록 다이어그램이다. 머신(10)에는 마스터 프로세서(12) 및 버스(16)를 통해 상기 마스터 프로세서와 상호 통신을 하는 코프로세서(14_1-14_n), 원격 장치(도 1에는 도시하지 않음)로부터 원 데이터(raw data)를 수신하는 입력 포트(18), 및 처리된 데이터를 상기 원격 소스로 제공하는 출력 포트(20)가 포함되어 있다. 상기 머신(10)에는 또한 마스터 프로세서(12)용 메모리(22), 코프로세서(14_1-14_n)용 메모리(24_1-24_n), 및 상기 버스(16)를 통해 마스터 프로세서와 코프로세서가 공유하는 메모리(26)도 포함되어 있다. 메모리(22)는 마스터 프로세서(12)를 위한 프로그램 및 작업 메모리 모두의 역할을 하고, 메모리(24_1-24_n) 각각은 코프로세서(14_1-14_n) 각각을 위한 프로그램 및 작업 메모리 모두의 역할을 한다. 공유 메모리(26)는 마스터 프로세서(12)와 코프로세서(14)가 그들 사이의 데이터를 포트(18 및 20)을 통해 각각 원격 장치로/장치로부터 전달할 수 있게 해준다. 마스터 프로세서(12) 및 코프로세서(14)는 또한 머신(10)이 원 데이터를 처리하는 속도를 제어하는 공통의 클록 신호를 수신하기도 한다.

일반적으로, 컴퓨팅 머신(10)은 마스터 프로세서(12)와 코프로세서(14) 간의 원 데이터 처리를 효과적으로 분배한다. 소나 어레이(sonar array)(도 5)와 같은 원격 소스(도 1에는 도시하지 않음)는 포트(18)를 통해 원 데이터를 상기 원 데이터를 위한 선입선출(FIFO) 버퍼(도시하지 않음)로서 작동하는 공유 메모리(26)의 일부에 로드한다. 마스터 프로세서(12)는 버스(16)를 통해 메모리(16)로부터 원 데이터를 검색하고, 마스터 프로세서와 코프로세서(14)가 상기 원 데이터를 처리하고, 버스(16)를 통해 필요한 만큼 그들사이에서 데이터를 전달한다. 마스터 프로세서(12)는 처리된 데이터를 공유 메모리(26)에 정의되어 있는 다른 FIFO 버퍼(도시하지 않음)에 로드하고, 원격 소스가 포트(20)를 통해 이 FIFO 로부터 상기 처리된 데이터를 검색한다.

다른 연산의 예에서, 컴퓨팅 머신(10)은 원 데이터를 처리하는데 있어서 상기 원 데이터상에서 각각의 연산에 n+1을 순차적으로 수행하여 처리하는데, 이 연산은 패스트 푸리에 변환(FFT)과 같은 처리 알고리즘을 함께 구성한다. 보다 특별하게는, 머신(10)은 마스터 프로세서(12)와 코프로세서(14)로부터 데이터-처리 파이프라인을 형성한다. 주어진 광폭 신호의 주파수를 위해, 그러한 파이프라인은 종종 머신(10)이 하나의 프로세서만 가지는 머신보다 더욱 빠르게 원 데이터를 처리할 수 있게 한다.

메모리(26)내의 원-데이터 FIFO(도시하지 않음)로부터 원 데이터를 검색한 후, 마스터 프로세서(12)는 삼각 함수와 같은 제1 연산을 상기 원 데이터상에 수행한다. 이 연산은 프로세서(12)가 메모리(26) 내부에 정의된 제1-결과 FIFO(도시하지 않음)내에 저장하는 첫번째 결과를 산출해 낸다. 일반적으로, 프로세서(12)는 메모리(12) 내에 저장된 프로그램을 수행하며, 그 프로그램의 제어하에 상기 설명된 연산들을 수행한다. 프로세서(12)는 또한 메모리(22)를 작업 메모리로 사용하여 프로세서가 상기 제1 연산의 중간 시간에 생성하는 데이터를 일시적으로 저장하기도 한다.

다음으로, 상기 메모리(26)내의 제1-결과 FIFO(도시하지 않음)로부터 첫번째 결과를 검색한 후, 코프로세서(14)는 로그 함수와 같은 두번째 연산을 상기 첫번째 결과상에 수행한다. 이 두번째 연산은 코프로세서(14_1)가 메모리(26) 내부에 정의된 제2-결과 FIFO(도시하지 않음)내에 저장하는 두번째 결과를 산출해 낸다. 일반적으로, 코프로세서(14_1)는 메모리(24_1) 내에 저장된 프로그램을 수행하며, 그 프로그램의 제어하에 상기 설명된 연산들을 수행한다. 코프로세서(14)는 또한 메모리(24_1)를 작업 메모리로 사용하여 코프로세서가 상기 제2 연산의 중간 시간에 생성하는 데이터를 일시적으로 저장하기도 한다.

그리고 나서, 코프로세서(24_2-24_n)는 상기 두번째-(n-1)번째 상에 세번째-n번째 연산을 순차적으로 수행하여 상기 코프로세서(24_1)를 위해 상기 설명한 것과 유사한 방법의 결과를 가져온다.

코프로세서(24_n)에 의해 수행되는 n번째 연산은 최종 결과, 즉 처리된 데이터를 산출한다. 코프로세서(24_n)는 메모리(26) 내부에 정의된 처리된-데이터 FIFO(도시하지 않음)로 이 처리된 데이터를 로드(load)하고, 원격 장치(도 1에는 도시하지 않음)가 이 FIFO로부터 상기 처리된 데이터를 검색한다.

마스터 프로세서(12)와 코프로세서(14)가 처리 알고리즘의 다른 연산을 동시에 수행하기 때문에, 컴퓨팅 머신(10)은 서로 다른 연산을 순차적으로 수행하는 하나의 프로세서를 갖는 컴퓨팅 머신 보다도 원 데이터를 보다 빨리 처리할 수 있기도 하다. 특히, 하나의 프로세서는 원 데이터의 앞 세트상에 모든 n+1 연산을 수행할 때 까지는 원 데이터의 새로운 세트를 검색할 수 없다. 그러나, 상기 언급한 파이프라인 기술을 이용해서, 마스터 프로세서(12)는 오직 첫번째 연산을 수행한 후 원 데이터의 새로운 세트를 검색할 수 있다. 따라서, 주어진 광폭 주파수를 위해, 이 파이프라인 기술은 머신(10)이 원 데이터를 처리하는데 있어서 하나의 프로세서 머신(도 1에는 도시하지 않음)과 비교할 때 대략 n+1 팩터 만큼 원 데이터를 처리하는 속도를 증가시킬 수 있다.

선택적으로, 컴퓨팅 머신(10)은 원 데이터상에, FFT 와 같은 처리 알고리즘의 경우에 n+1을 동시에 수행함으로써 병렬로 원 데이터를 처리할 수도 있다. 즉, 알고리즘에 앞서의 실시예에서 상기 언급한 바와 같은 n+1 순차적 연산이 포함되어 있다면, 마스터 프로세서(12)와 코프로세서(14) 각각은 모든 n+1 연산을 원 데이터 각각의 세트상에서 수행한다. 따라서, 주어진 광폭 주파수를 위해서는, 상기 설명한 파이프라인 기술과 같이, 이러한 병렬-처리 기술은 머신(10)이 하나의 프로세서 머신(도 1에는 도시하지 않음)과 비교할 때 대략 n+1 팩터 만큼 원 데이터 처리 속도를 증가시킬 수 있다.

불행하게도, 비록 컴퓨팅 머신(10)이 하나의 프로세서 컴퓨터 머신(도 1에는 도시하지 않음)보다 빨리 데이터를 처리할 수는 있으나, 머신(10)의 데이터-처리 속도가 종종 프로세서 광폭의 주파수보다 적어지곤 한다. 특히, 컴퓨팅 머신(10)의 데이터-처리 속도는 마스터 프로세서(12)와 코프로세서(14)가 데이터를 처리하는데 요구하는 시간에 의해 제한된다. 간

확히 하기 위해, 이 속도 제한의 한 예를 마스터 프로세서(12)를 참고하여 설명하는데, 이 설명은 코프로세서(14)에도 적용됨을 이해할 수 있을 것이다. 앞에서 설명한 바와 같이, 마스터 프로세서(12)는 프로세서를 제어하여 데이터를 원하는 방식으로 조작하도록 제어하는 프로그램을 실행한다. 이 프로그램에는 프로세서(12)가 실행하는 일련의 명령이 포함되어 있다. 통행하게도, 프로세서(12)는 일반적으로 하나의 명령을 수행하는데 다중 클록 사이클을 필요로 하는데, 종종 다중 명령을 수행하여 데이터의 하나의 값을 처리해야 한다. 예를 들어, 프로세서(12)가 제1 데이터 값(A)(도시하지 않음)과 제2 데이터 값(B)(도시하지 않음)을 곱하는 경우를 가정한다. 제1 클록 사이클 동안, 프로세서(12)는 메모리(22)로부터 여러개의 명령을 검색한다. 제2 및 제3 클록 사이클 동안, 프로세서(12)는 메모리(22)로부터 A와 B를 각각 검색한다. 제4 클록 사이클 동안, 프로세서(12)는 A와 B를 곱하고, 제5 클록 사이클 동안, 그 결과물을 메모리(22 또는 26)에 저장하거나 또는 그 결과물을 원격 장치(도시하지 않음)로 제공한다. 이것은 최선의 시나리오인데, 그 이유는 많은 경우에서, 프로세서(12)는 카운터를 초기화(initializing) 및 닫는(closing) 경우와 같이 오버헤드 태스크(overhead task)를 위한 추가의 클록 사이클을 요구하기 때문이다. 그러므로, 프로세서(12)는 A와 B를 처리하기 위해서는 5개의 클록 사이클, 또는 데이터 값 당 평균 2.5개의 클록 사이클을 필요로 한다.

따라서, 컴퓨팅 머신(10)이 데이터를 처리하는 속도는 종종 마스터 프로세서(12) 및 코프로세서(14)를 구동하는 클록의 주파수보다 크게 낮아지곤 한다. 예를 들어, 프로세서(12)가 1.0 기가헤르쯔(GHz)에서 클록되지만 데이터 값 당 평균 2.5 클록 사이클을 요구한다면, 유효 데이터-처리 속도는 (1.0GHz)/2.5=4GHz 가 될 것이다. 이 유효 데이터-처리 속도는 종종 초당 연산 단위로 특정되곤 한다. 그러므로, 1.0GHz 의 클록 속도를 위해서는, 프로세서(12)는 초당 0.4 기가연산(Gops)의 데이터-처리 속도로 레이트(rate)되어야 한다.

도 2는 프로세서가 주어진 클록 주파수 및 종종 파이프라인이 클록되는 레이트와 거의 등일한 레이트에서 할 수 있는 것보다 더 빠른 일반적인 데이터를 처리할 수 있는 하드와이어드(hardwired) 데이터 파이프라인(30)의 블록 다이어그램이다. 파이프라인(30)에는 각각 실행 프로그램 명령 없이 각자의 데이터상의 개별적 연산을 각각 수행하는 연산자 회로(32_1-32_n)가 포함되어 있다. 즉, 원하는 연산이 회로(32)로 "번 되는(burned in)" 것으로 프로그램 명령 없이도 자동적으로 연산을 실행하는 것이다. 실행 프로그램 명령과 관련된 오버헤드를 제어함으로써, 파이프라인(30)은 종종 주어진 클록 주파수를 위해 할 수 있는 것보다 더 많은 연산을 수행할 수 있다.

예를 들어, 파이프라인(30)은 프로세서가 주어진 클록 주파수를 위해 할 수 있는 것보다 더 빠른 다음과 같은 수식을 풀 수 있다.

$$Y(X_k)=(5X_k+3)2^{\propto k}$$

여기서, Xk는 일련의 원 데이터 값을 나타낸다. 이 예에서, 연산자 회로(32_1)는 5Xk 를 계산하는 곱셈기이고, 회로(32_2)는 5Xk + 3 을 계산하는 가산기이며, 회로(32_n)(n=3)는 (5Xk + 3)$2^{\propto k}$ 를 계산하는 곱셈기이다.

제1 클록 사이클 k=1 동안, 회로(32_1)는 데이터 값(X_1)을 수신하고 여기에 5를 곱해 $5X_1$ 을 생성한다.

제2 클록 사이클 k=2 동안, 회로(32_2)는 회로(32_1)로부터 $5X_1$을 수신하고, 3을 더해서 $5X_1$ + 3 을 생성한다. 또한, 상기 제2 클록 사이클 동안, 회로(32_1)는 $5X_2$ 를 생성한다.

제3 클록 사이클 k=3 동안, 회로(32_3)는 회로(32_2)로부터 $5X_1$ + 3 을 수신하고, 2^{X1} (x1 만큼 유효하게 $5X_1$ + 3 오른쪽 시프트)를 곱하여 첫번째 결과($5X_1$ + 3)2^{x1} 을 생성한다. 또한, 제3 클록 사이클 동안, 회로(32_1)는 $5X_3$ 를 생성하고 회로(32_2)는 $5X_2$ + 3 을 생성한다.

파이프라인(30)은 이러한 방식으로 모든 원 데이터 값이 처리될 때 까지 연속적인 원 데이터 값(X_k)을 계속 처리한다.

따라서, 한 데이터 값(X_1)을 수신한 후 두 개의 클록 사이클의 지연 - 이 지연을 파이프라인(30)의 레이턴시(latency)라고 부르기로 함 - 상기 파이프라인은 결과 $(5X_1 + 3)2^{X1}$ 을 생성하고, 그 후에 하나의 결과를 각각의 클록 사이클을 생성한다.

상기 레이턴시를 무시하면, 파이프라인(30)은 클록 속도와 동일한 데이터-처리 속도를 갖는다. 비교를 하면, 마스터 프로세서(12)와 코프로세서(14)가 상기 예에서와 같은 클록 속도의 0.4배의 데이터-처리 속도를 갖는다고 가정하면, 파이프라인(30)은 주어진 클록 속도를 위해 컴퓨팅 머신(10)(도 1)보다 2.5배 빨리 데이터를 처리할 수 있다.

도 2를 계속 참조하면, 설계자는 파이프라인(30)을 필드-프로그램가능한 게이트 어레이(FPGA)와 같은 프로그램가능한 로직 IC(PLIC)에서 수행하도록 선택하기도 하는데, 그 이유는 PLIC 는 주문형 반도체(ASIC)보다 나은 디자인 및 변형 유용성 가진다. PLIC 내부에서 하드와이어드 접속을 구성하기 위해서는, 설계자는 단지 PLIC 내부에 배치된 상호접속-구성 레지스터를 미리 결정된 이진 상태(binary stste)로 설정하기만 하면 된다. 이들 이진 상태 모두의 조합을 "펌웨어(firmware)"라고 부르기로 한다. 일반적으로, 설계자는 이 펌웨어를 PLIC 와 관련된 비휘발성 메모리(도 2에 도시하지 않음)에 프로그램한다. 누군가가 PLIC 를 "켜면(turn on)", PLIC는 상기 메모리로부터 펌웨어를 상기 상호접속-구조 레지스터로 다운로드한다. 그러므로, PLIC 의 기능을 변경시키기 위해서는, 설계자는 단지 펌웨어를 수정하면 되고 PLIC 로 하여금 그 수정된 펌웨어를 상호접속-구조 레지스터로 다운로드하게 하면 된다. 이것은 단지 펌웨어를 수정하는 것에 의해 PLIC를 수정할 수 있다는 것은 시제품화 단계 동안 및 "필드 내에서" 파이프라인(30)의 업그레이드를 위해 특히 유용하다.

불행하게도, 하드와이어드 파이프라인(30)은 중요한 결정 형성을 필요로 하는 모든 알고리즘을 실행하지 못한다. 프로세서는 비교 가능한 길이의 연산 명령(예를 들어, "A+B")를 실행할 수 있는 정도로 거의 빠르게 일반적으로 결정-형성 명령(예를 들어, "A이면 B로 가고, 그렇지 않고 C로 가고"와 같은 조건 명령)을 실행할 수 있다. 그러나, 비록 파이프라인(30)이 상대적으로 간단한 결정(예를 들어, "A>B?")을 구성할 수 있어도, 일반적으로는 상대적으로 복잡한 결정(예를 들어, "A이면 B로 가고, 그렇지 않으면 C로 가고")을 실행하지 못한다. 그리고, 비록 그러한 복잡한 결정을 실행하기 위해 파이프라인(30)을 디자인할 수 있다 하여도, 요구되는 회로의 크기와 복잡성은 종종 설계를 불가능하게 만드는데, 특히 여러개의 서로 다른 복잡한 결정을 포함하는 알고리즘인 경우 그러하다.

따라서, 프로세서는 중요한 결정 형성을 요구하는 애플리케이션 내에서 종종 사용되며, 하드와이어드 파이프라인은 결정 형성이 거의 없는 또는 전혀 없는 "수치처리(number crunching)" 애플리케이션으로 제한되곤 한다.

더욱이, 아래에 설명한 바와 같이, 도 2의 파이프라인(30)과 같은 하드와이어드 파이프라인, 특히 여러개의 PLIC를 포함하는 파이프라인(30)을 설계/수정하는 것 보다는 도 1의 컴퓨팅 머신(10)과 같은 프로세서-기반 컴퓨팅 머신을 설계/수정하는 것이 훨씬 쉽다.

프로세서와 그 주변장치들(예를 들어, 메모리)과 같은 컴퓨팅 구성요소들은 일반적으로 이 구성요소들의 상호접속을 촉진하여 프로세서-기반 컴퓨팅 머신을 형성하기 위한 산업-표준 통신 인터페이스를 포함한다.

특히, 표준 통신 인터페이스에는 두 개의 계층(layer)이 포함되는데, 물리 계층과 서비스 계층이다. 물리 계층에는 회로 및 이 회로의 연산 파라메터 및 인터페이스를 형성하는 대응 회로 상호접속이 포함되어 있다. 예를 들어, 물리 계층에는 구성요소를 버스와 연결하는 핀, 이 핀으로부터 데이터를 래치(latch)하는 버퍼, 및 이 핀상에서 데이터를 구동시키는 구동기가 포함되어 있다. 상기 연산 파라메터에는 상기 핀이 수신하는 데이터 신호의 허용가능한 전압 범위, 데이터를 기록 및 판독하는 신호 타이밍, 및 지지된 연산 모드(예를 들어, 버스트 모드, 페이지 모드)가 포함되어 있다. 종래의 물리 계층에는 트랜지스터-트랜지스터 논리(TTL) 및 램버스(RAMBUS)가 포함된다.

서비스 계층에는 컴퓨팅 구성요소가 데이터를 전송하는 프로토콜이 포함된다. 이 프로토콜은 데이터의 포맷 및 상기 구성요소가 포맷된 데이터를 송수신하는 방식을 정의한다. 종래의 통신 프로토콜에는 파일-전송 프로토콜(FTP) 및 확장이 포함된다.

따라서, 산업-표준 통신 계층을 갖는 제조자 및 다른 일반적인 설계 컴퓨팅 구성요소들로 인해서, 그러한 구성요소의 인터페이스를 일반적인 설계로 할 수 있고 이것을 상대적으로 적은 노력으로 다른 컴퓨팅 구성요소들과 상호접속시킬 수 있다. 이것은 설계자에게 대부분의 시간을 컴퓨팅 머신의 다른 부분들을 설계하는데 소비하게 만들고, 구성요소들을 추가하거나 없애는 것을 통해 머신을 쉽게 수정할 수 있게 한다.

산업-표준 통신 계층을 지원하는 컴퓨팅 구성요소를 설계하는 것은 설계 라이브러리(design library)로부터 현존하는 물리-계층 설계를 사용하여 설계 시간을 절약하게 해 준다. 이것은 또한 구성요소들은 재고품인 컴퓨팅 구성요소들과 쉽게 접속할 수 있게 해주기도 한다.

공통의 산업-표준 통신 계층을 지원하는 컴퓨팅 구성요소를 사용하여 컴퓨팅 머신을 설계하는 것은 설계자로 하여금 시간과 노력을 줄여주면서 구성요소들을 상호접속할 수 있게 해 준다. 이들 구성요소들이 공통의 인터페이스 계층을 지원하기 때문에, 설계자는 설계 노력을 거의 들이지 않고 시스템 버스를 통해 이들을 상호 접속할 수 있다. 지원되는 인터페이스 계층이 산업 표준이기 때문에, 설계자는 머신을 쉽게 수정할 수 있다. 예를 들어, 설계자는 다른 구성요소 및 주변장치들을 머신에 추가하여 시스템 설계를 발전시켜 나갈 수 있으며, 또는 차세대 구성요소들을 쉽게 추가/설계하여 기술 발전을 이룰 수 있다. 더욱이, 구성요소들이 공통의 산업-표준 서비스 계층을 지원하기 때문에, 설계자는 컴퓨팅 머신의 소프트웨어로 대응하는 프로토콜을 실현하는 현존하는 소프트웨어 모듈을 합체시킬 수 있다. 따라서, 설계자는 인터페이스 설계가 이미 적절하게 원숙적이기 때문에 거의 노력을 들이지 않고 구성요소들을 접속시킬 수 있어서 머신이 특정 기능을 수행하게 하는 머신의 일부(예를 들어, 소프트웨어)를 설계하는 데 주력할 수 있다.

그러나, 불행하게도, 도 2의 파이프라인(30)과 같은 하드와이어드 파이프라인을 형성하는데 사용되는 PLIC 등과 같은 구성요소를 위한 알려진 산업-표준 통신 계층은 없다.

따라서, 여러개의 PLIC 를 갖는 파이프라인을 설계하기 위해서, 설계자는 "스크래치(scratch)로부터" PLIC 간의 통신 계층을 설계하고 디버깅하는데 상당한 시간과 노력을 들여야 한다. 일반적으로, ad hoc 통신 계층은 PLIC 간에서 전달되는 데이터의 파라메터에 따라 달라진다. 비슷하게, 프로세서와 접속되는 파이프라인을 설계하기 위해서는, 설계자는 스크래치로부터 파이프라인과 프로세서간의 통신 계층을 설계하고 디버깅하는데 상당한 시간과 노력을 들여야 한다.

비슷하게, PLIC 을 추가하는 것으로 파이프라인을 수정하기 위해서는, 설계자는 일반적으로 추가된 PLIC과 현재의 PLIC 사이의 통신 계층을 설계하고 디버깅하는데 상당한 시간과 노력을 들이게 된다. 그리고, 프로세서를 추가하는 것으로 파이프라인을 수정하려면, 또는, 파이프라인을 추가하는 것으로 컴퓨팅 머신을 수정하려면, 설계자는 파이프라인과 프로세서간의 통신 계층을 설계하고 디버깅하는데 상당한 시간과 노력을 들여야 한다.

그러므로, 도 1 및 도 2를 참고하면, 다수의 PLIC 를 접속하고 파이프라인과 프로세서의 접속 어려움으로 인해, 설계자는 컴퓨팅 머신을 설계하는 경우 상당한 트레이드오프(tradeoff)에 직면하곤 한다. 예를 들어, 프로세서-기반 컴퓨팅 머신을 가지고는, 설계자는 복잡한 결정-형성 가능성을 위한 트레이드 수-코딩칭 속도 및 설계/수정 유연성에 집중하게 된다. 반대로, 하드와이어드 파이프라인-기반 컴퓨팅 머신을 가지고는, 설계자는 수-코딩칭 속도를 위한 트레이드 복잡성-결정-형성 가능성 및 설계/수정에 집중하게 된다. 더욱이, 다수의 PLIC 를 접속하는데의 어려움으로 인해, 소수의 PLIC 를 가지는 파이프라인-기반 머신을 설계하는 것이 불가능하기도 하다. 그 결과, 실제적인 파이프라인-기반 머신은 제한된 기능을 갖곤 한다. 그리고, 프로세서와 PLIC 와의 접속 어려움으로 인해서, 하나의 PLIC 이상과 프로세서와의 접속이 불가능하기도 하다. 그 결과, 프로세서와 파이프라인을 참짐으로써 얻어지는 이익이 적다.

따라서, 하드와이어드-파이프라인-기반 머신의 수-코딩칭 속도를 가지고 프로세서-기반 머신의 결정-형성 가능성을 결합시킬 수 있는 새로운 컴퓨터 아키텍처 요구가 있어 왔다.

발명의 상세한 설명

(요약)

본 발명의 한 실시예에서, 피어-벡터 머신에는 호스트 프로세서 및 하드와이어드-파이프라인 가속기가 포함되어 있다. 상기 호스트 프로세서는 프로그램을 실행하고, 그 프로그램에 응답하여, 호스트 데이터를 생성하고, 상기 파이프라인 가속기는 상기 호스트 데이터로부터 파이프라인 데이터를 생성한다.

본 발명의 다른 실시예에 따르면, 상기 파이프라인 가속기는 상기 파이프라인 데이터를 생성하고, 상기 호스트 프로세서는 상기 파이프라인 데이터로부터 상기 호스트 데이터를 생성한다.

상기 피어-벡터 머신에 프로세서와 하드와이어드 가속기가 모두 포함되어 있기 때문에, 프로세서만 포함하는 또는 하드와이어드 파이프라인만 포함하는 컴퓨팅 머신보다 효과적으로 데이터 처리를 할 수 있다. 예를 들어, 설계자는 상기 호스트 프로세서가, 상기 가속기가 수학적으로 집중 연산을 수행하는 동안, 결정-형성 및 비-수학적 집중 연산을 수행하도록 피어 벡터 머신을 설계할 수 있다.

도면의 간단한 설명

도 1은 종래의 다중-프로세서 아키텍처를 갖는 컴퓨팅 머신의 블럭 다이어그램이고,

도 2는 종래의 하드와이어드 파이프라인의 블럭 다이어그램이고,

도 3은 본 발명의 일 실시예에 따른 피어-벡터 아키텍처를 갖는 컴퓨팅 머신의 개략적인 블럭 다이어그램이고,

도 4는 본 발명의 일 실시예에 따른 도 3의 피어-벡터 컴퓨팅 머신을 통합하는 전자 시스템의 개략적 블럭 다이어그램이다.

실시예

(상세한 설명)

도 3은 본 발명의 일 실시예에 따른 피어-벡터 아키텍처를 갖는 컴퓨팅 머신(40)의 개략적인 블럭 다이어그램이다. 호스트 프로세서(42)에 추가하여, 상기 피어-벡터 머신(40)에는 적어도 일부의 데이터 처리를 수행하여 도 1의 컴퓨팅 머신(10) 내의 코프로세서(14)의 뱅크(bank)를 유효하게 대체하는 파이프라인 가속기(44)가 포함되어 있다. 따라서, 호스트-프로세서(42) 및 가속기(44)는 데이터 벡터를 양방향 전송할 수 있는 "피어(peer)" 이다. 가속기(44)는 프로그램 명령을 실행하지 않으므로, 가속기는 종종 코프로세서의 뱅크가 주어진 클럭 주파수에서 할 수 있는 것보다 훨씬 빠르게 데이터상의 집중적인 연산을 수학적으로 처리한다. 따라서, 프로세서(42)의 결정-형성 가능성과 가속기(44)의 수-크런칭 가능성을 결합함으로써, 머신(40)은 동일한 능력을 갖지만, 머신(10)과 같은 종래의 컴퓨팅 머신보다는 빠르게 데이터를 처리할 수 있다. 또한, 앞서 언급한 발명의 명칭이 "COMPUTING MACHINE HAVING IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD" 인 미국 특허출원 제10/684,053호 및 발명의 명칭이 "PIPELINE ACCELERATOR FOR IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD" 인 미국 특허출원 제10/683,929호에 설명된 바와 같이, 가속기(44)에 상기 호스트 프로세서(42)의 통신 계층을 제공하는 것은 머신(40)의 설계 및 수정을 용이하게 해 주고, 특히, 프로세서의 통신 계층이 산업 표준인 경우 그러하다. 가속기(44)에 다수의 구성요소(예를 들어, PLIC)이 포함되어 있는 경우, 이들 구성요소에 동일한 계층을 제공하는 것은 가속기의 설계 및 수정을 용이하게 해주는데, 특히 상기 통신 계층이 산업-표준과 호환되는 경우 그러하다. 더욱이, 머신(40)은 후술하는 바와 같은 그리고 앞서 언급한 다른 출원들에서의 다른 장점도 제공한다.

호스트 프로세서(42) 및 파이프라인 가속기(44)에 추가하여, 피어-벡터 컴퓨팅 머신(40)에는 프로세서 메모리(46), 인터페이스 메모리(48), 버스(50), 펌웨어 메모리(52), 선택적인 원-데이터 입력 포트(54,56), 처리된-데이터 출력 포트(58,60), 및 선택적인 라우터(61)가 포함되어 있다.

호스트 프로세서(42)에는 처리 유닛(62) 및 메시지 처리기(64)가 포함되어 있으며, 프로세서 메모리(46)에는 처리-유닛 메모리(66) 및 처리기 메모리(68)을 포함하는데, 각각 프로세서 유닛 및 메시지 처리기를 위한 프로그램 및 작업 메모리 모두의 역할을 한다. 프로세서 메모리(46)에는 가속기-구성 레지스트리(70) 및 메시지-구성 레지스트리(72)도 포함되어 있는데, 이들은 각각 호스트 프로세서(42)로 하여금 가속기(44)의 기능 및 메시지 처리기(64)가 생성하는 메시지의 구조를 구성하도록 하는 각각의 구성 데이터를 저장하고 있다.

파이프라인 가속기(44)는 적어도 하나의 PLIC(도시하지 않음)에 배치되어 있으며 프로그램 명령을 실행하지 않고 각각의 데이터를 처리하는 하드와이어드 파이프라인(74_1-74_n)을 포함하고 있다. 펌웨어 메모리(52)는 가속기(44)용 구성 펌웨어를 저장한다. 만일 가속기(44)가 다수의 PLIC 에 배치된다면, 이들 PLIC 및 그들 각각의 펌웨어 메모리는 다수의 회로 보드, 즉 미터 카드(daughter card)(도시하지 않음)상에 배치된다. 상기 가속기(44)와 미터 카드는 앞서 언급한 발명의 명칭이 "PIPELINE ACCELERATOR FOR IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD"인 미국 특허출원 제10/683,929호 및 발명의 명칭이 "PIPELINE ACCELERATOR HAVING MULTIPLE

PIPELINE UNITS AND RELATED COMPUTING MACHINE AND METHOD"인 미국 특허출원 제10/683,932호에 설명되어 있다. 선택적으로, 상기 가속기(44)는 적어도 하나의 ASIC 에 배치될 수 있으며, 따라서, 구성할 수 없는 내부 상호접속을 갖는다. 이 대안에서, 머신(40)에서 펌웨어 메모리(52)를 생략할 수 있다. 또한, 비록 가속기(44)가 다중 파이프라인(74)을 포함하는 것으로 도시되어 있으나, 오직 하나의 파이프라인만 포함할 수 있다. 또한, 비록 도시하지는 않았지만, 가속기(44)에는 디지털-신호 처리기(DSP)와 같은 하나 또는 그 이상의 프로세서가 포함될 수 있다.

도 3을 계속 참조하여, 파어-벡터 머신(40)의 동작은 본 발명의 일 실시예에 따라 후술한다.

파어-벡터 머신 구성

파어-벡터 머신(40)이 가장 먼저 활성화 되면, 처리 유닛(62)은 메시지 처리기(64) 및 파이프라인 가속기(44)(상기 가속기는 구성가능함)를 구성하여 머신이 원하는 알고리즘을 실행할 수 있도록 한다. 특히, 처리 유닛(62)은 메모리(66)내에 저장되어 있고 상기 처리 유닛으로 하여금 메시지 처리기(164)와 가속기(44)를 후술하는 바와 같이 구성하도록 만드는 호스트 애플리케이션 프로그램을 실행한다.

메시지 처리기(64)를 구성하기 위해서는, 처리 유닛(62)은 레지스트리(72)에서 메시지-포맷 정보를 검색하고 이 포맷 정보를 메모리(60)에 이 정보를 저장하는 메시지 처리기로 제공한다. 머신(40)이 후술하는 바와 같이 데이터를 처리하면, 메시지 처리기(64)는 이 포맷 정보를 사용하여 원하는 포맷을 갖는 데이터 메시지를 생성하고 판독한다. 한 실시예에서, 상기 포맷 정보는 확장성 생성 언어(XML)로 기록되며, 다른 언어나 다른 데이터 포맷으로 기록될 수도 있다. 상기 처리 유닛(62)이 상기 메시지 처리기(64)를 상기 파어-벡터 머신(40)이 활성화 되는 시간마다 구성하기 때문에, 설계자는 레지스트리(72)에 저장되어 있는 포맷 정보를 단지 수정하는 것으로 메시지 포맷을 수정할 수 있다. 선택적으로, 외부 메시지-구성 라이브러리(도시하지 않음)는 다중 메시지 포맷을 위한 정보를 저장할 수 있고, 설계자는 상기 처리 유닛(62)이 상기 라이브러리의 선택된 부분에서 레지스트리(72)를 업데이트하여 그 업데이트된 레지스트리로부터 상기 메시지 처리기(64)로 원하는 포맷 정보를 다운로드할 수 있도록 상기 호스트 애플리케이션을 설계 및/또는 수정할 수 있다. 상기 메시지 포맷 및 메시지를 생성하고 판독하는 것은 후술하며, 상기 언급한 발명의 명칭이 "COMPUTING MACHINE HAVING IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD" 인 미국 특허출원 제10/684,053호에 언급되어 있다.

유사하게, 파이프라인 가속기(44)의 상호접속 레이아웃을 구성하기 위해서는, 처리 유닛(62)은 상기 레지스트리(70)로부터 구성 펌웨어를 검색하고 이 펌웨어를 상기 메시지 처리기(64) 및 버스(50)를 통해 메모리(52)로 다운로드 한다. 그리고 나서 가속기(44)는 상기 메모리(52)로부터 상기 펌웨어를 다운로드함으로써 그 자신을 상호접속-구성 레지스터(도시하지 않음)로 구성한다. 처리 유닛(62)이 상기 가속기(44)를 상기 파어-벡터 머신(40)이 활성화되는 시간 마다 구성하기 때문에, 설계자는 상기 레지스트리(70)내에 저장된 펌웨어를 수정하는 것만으로 가속기(44)의 상호접속 레이아웃 - 기능화(functioning) - 을 수정할 수 있다. 선택적으로, 외부 가속기-구성 라이브러리(도시하지 않음)는 가속기(44)의 다중 구성을 위한 펌웨어를 저장할 수 있고, 설계자는 상기 처리 유닛(62)이 상기 라이브러리의 선택된 부분에서 상기 레지스트리(70)를 업데이트하여 상기 업데이트된 레지스트리로부터 상기 메모리(52)로 원하는 펌웨어를 다운로드할 수 있도록 상기 호스트 애플리케이션을 설계 및/또는 수정할 수 있다. 또한, 외부 라이브러리 또는 레지스트리(70)는 상기 가속기(44)의 서로 다른 부분 및/또는 기능을 정의하는 펌웨어 모듈을 저장한다. 따라서, 설계자는 이들 모듈을 사용하여 가속기(44)의 설계 및/또는 수정을 용이하게 할 수 있다. 또한, 처리 유닛(62)이 이들 모듈을 사용하여 상기 머신(40)이 데이터를 처리하는 동안 가속기(44)를 수정한다. 상기 가속기(44) 및 펌웨어의 상호접속-구성은 앞서 언급한 발명의 명칭이 "PROGRAMMABLE CIRCUIT AND RELATED COMPUTING MACHINE AND METHOD" 인 미국 특허출원 제10/684,057호에 더 설명되어 있다.

[54]~[56]

상기 처리 유닛(62)은, 상기 파어-벡터 머신(40)이 데이터를 처리하는 동안 파이프라인 가속기(44)를 "소프트 구성" 하기도 한다. 즉, 처리 유닛(62)은 가속기의 상호접속 레이아웃을 변경하지 않고 가속기(44)의 기능을 구성한다. 그러한 소프트 구성은 후술되어 있으며 상기 언급한 발명의 명칭이 "PIPELINE ACCELERATOR FOR IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD" 인 미국 특허출원 제10/683,929호에 더 설명되어 있다.

파어-벡터 머신으로 데이터 처리

일반적으로, 피어-백터 머신(40)은 호스트 프로세서(42)와 파이프라인 가속기(44) 간의 원 데이터의 처리를 효과적으로 분할한다. 예를 들어, 호스트 프로세서(42)는 상기 데이터와 관련된 결정-형성 연산의 대부분 또는 모두를 수행하고, 가속기(44)는 상기 데이터상의 수학적인 집중 연산의 대부분 또는 모두를 수행한다. 그러나, 머신(40)은 이 데이터 처리를 어떠한 원하는 방법으로 분할할 수 있다.

호스트 프로세서의 동작

한 실시예에서, 호스트 프로세서(42)는 원 데이터를 수신하고 그 결과인 처리된 데이터를 소나 어레이와 같은 원격 장치로 제공한다.

호스트 프로세서(42)는 우선 입력 포트(54) 또는 버스(50)를 통해 상기 원격 장치로부터 상기 원 데이터를 수신한다. 피어-백터 머신(40)에는 상기 수신된 원 데이터를 버퍼링하기 위한 FIFO(도시하지 않음)가 포함되어 있다.

다음으로, 처리 유닛(62)은 파이프라인 가속기(44)에 의한 처리를 위한 상기 원 데이터를 준비한다. 예를 들어, 상기 유닛(62)은 어느 원 데이터를 가속기(44)로 전송할지 또는 어느 시퀀스를 가지고 상기 원 데이터를 전송할지를 결정한다. 또는, 상기 유닛(62)은 상기 원 데이터를 처리하여 가속기(44)로 전송하기 위한 중간 데이터를 생성한다. 상기 원 데이터의 준비에 대해서는 앞서 언급한 발명의 명칭이 "COMPUTING MACHINE HAVING IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD" 인 미국 특허출원 제10/684,053호에 더 설명되어 있다.

원 데이터를 준비하는 동안, 처리 유닛(54)은 가속기(44)의 기능을 수정하기 위한 하나 또는 그 이상의 "소프트-구성" 명령을 생성하기도 한다. 머신(40)이 활성화 될 때 가속기(44)의 상호접속 레이아웃을 구성하는 펌웨어와는 달리, 소프트-구성 명령은 그 상호접속 레이아웃을 변경하지 않고 가속기의 구성을 제어한다. 예를 들어, 소프트-구성 명령은 가속기(44)가 처리하는 데이터 스트링(예를 들어, 32비트 또는 64비트)의 크기를 제어한다. 가속기(44)의 소프트 구성은 앞서 언급한 발명의 명칭이 "PIPELINE ACCELERATOR FOR IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD" 인 미국 특허출원 제10/683,929호에 더 설명되어 있다.

그러면 처리 유닛(62)은 준비된 데이터 및/또는 소프트-구성 명령을, 상기 유닛(62)과 상기 가속기(44) 사이에서 FIFO 버퍼로 동작하는 인터페이스 메모리(48)의 해당 위치로 포트한다.

다음으로, 메시지 처리기(64)는 상기 인터페이스 메모리(48)로부터 상기 준비된 데이터 및/또는 소프트웨어 명령을 검색하고, 상기 데이터 및/또는 명령 및 관련된 정보를 포함하는 메시지 오브젝트(message object)를 생성한다. 일반적으로, 가속기(44)는 상기 데이터/명령 및 관련된 정보를 설명하는 네 개의 식별기(집합적으로 "정보")를 필요로 하는데: a) 정보의 의도된 목적지(예를 들어, 파이프라인(74$_1$)), b) 우선권(예를 들어, 가속기가 앞서 수신된 데이터 전후로 이 데이터를 처리해야 하는지), c) 상기 메시지 오브젝트의 길이 또는 엔드(end), 및 d) 상기 데이터의 고유 인스턴스(instance)(예를 들어, 1000 개 센서의 어레이로부터 센서 신호 번호 9). 이 결정을 촉진하기 위해서, 메시지 처리기(164)는 상기 설명한 바와 같은 미리 결정된 포맷을 가지는 메시지 오브젝트를 생성한다. 준비된 데이터/소프트-구성 명령에 추가하여, 메시지 오브젝트에는 보통 네 개의 상기 언급한 식별기를 포함하고, 오브젝트가 포함하는 정보의 타입(예를 들어, 데이터, 명령)을 설명하는 식별기, 및 상기 데이터가 처리될 알고리즘을 포함하는 헤더가 포함되어 있다. 후자의 식별기는 목적지 파이프라인(74)이 다중 알고리즘을 수행하는 경우 유용하다. 상기 처리기(64)는 인터페이스 메모리(48)로부터 상기 헤더 정보를 검색하거나 또는 준비된 데이터 또는 명령을 검색하는 인터페이스 메모리 내부의 위치에 기초하여 상기 헤더를 생성한다. 상기 메시지 헤더를 판독함으로써, 라우터(61) 및/또는 가속기(44)는 상기 메시지 오브젝트 내부의 정보를 원하는 목적지로 향하게 할 수 있고, 그 목적지가 원하는 시퀀스로 상기 정보를 처리하도록 하게 한다.

상기 메시지 오브젝트를 생성하는 선택적 실시예가 있다. 예를 들어, 각각의 메시지 오브젝트가 데이터 또는 소프트-구성 명령 중 어느 하나를 포함하는 것으로 설명하였으나, 하나의 메시지 오브젝트에는 데이터와 하나 또는 그 이상의 명령이 모두 포함되어도 좋다. 또한, 상기 메시지 처리기(64)를 인터페이스 메모리(48)로부터 상기 데이터와 명령을 수신하는 것으로 설명하였으나, 처리 유닛(54)을 통해 직접 데이터와 명령을 수신해도 좋다.

메시지 오브젝트의 생성에 대해서는 앞서 언급한 발명의 명칭이 "COMPUTING MACHINE HAVING IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD" 인 미국 특허출원 제10/684,053호에 더 설명되어 있다.

파이프라인 가속기

파이프라인 가속기(44)는 메시지 처리기(64)로부터 메시지 오브젝트를 수신하고 판독하며 상기 오브젝트 내부의 데이터 및/또는 명령을 원하는 목적지로 유효하게 향하도록 한다. 이 기술은 상기 처리 유닛(62) 및 파이프라인(74)에 의해 수행되는 알고리즘의 수가 상대적으로 적은 경우 특히 유용해서, 라우터(61)를 생략할 수 있다. 선택적으로, 상기 처리 유닛(62)에 의해 수행되는 알고리즘의 수 또는 파이프라인(74)의 수가 상대적으로 큰 경우에는, 라우터(61)가 상기 메시지 처리기(64)로부터 메시지 오브젝트를 수신하고 판독하고 상기 오브젝트 내부의 데이터 및/또는 명령이 가속기(44) 내부의 원하는 목적지로 유효하게 향하도록 한다.

한 실시예에서, 처리-유닛 알고리즘 및 파이프라인(74)의 수가 적은 경우, 각각의 파이프라인은 메시지 오브젝트를 동시에 수신하고 상기 헤더를 분석하여 그것이 상기 메시지의 원하는 수신측인지 아닌지를 결정한다. 만일, 메시지 오브젝트가 특정 파이프라인(74)으로 의도된 것이라면, 그 파이프라인은 그 메시지를 판독하고 복구된 데이터/명령을 처리한다. 그러나, 만약 상기 메시지 오브젝트가 특정 파이프라인(74)으로 의도되지 않았다면, 그 파이프라인은 상기 메시지 오브젝트를 무시한다. 예를 들어, 메시지 오브젝트가 파이프라인(74_1)에 의해 처리되는 데이터가 포함되어 있다고 가정한다. 따라서, 상기 파이프라인(74_1)은 상기 메시지 헤더를 분석하고, 그것이 데이터를 위한 의도된 목적지임을 결정하고, 상기 메시지로부터 데이터를 복구하고, 상기 복구된 데이터를 처리한다. 반대로, 상기 파이프라인(74_1~74_n) 각각은 상기 메시지 헤더를 분석하고 그것이 상기 데이터를 위한 의도된 목적지가 아님을 결정해서, 그 데이터를 복구 또는 처리하지 않는다. 만일 상기 메시지 오브젝트 내부의 데이터가 다중 파이프라인(74)을 위한 것이라면, 메시지 처리기(64)는 동일한 데이터를 포함하는 각각의 메시지 오브젝트의 시퀀스, 각각의 목적지 파이프라인을 위한 하나의 메시지를 생성하고 전송한다. 선택적으로, 메시지 처리기(64)는, 목적지 파이프라인 모두를 식별하는 헤더를 가지는 하나의 메시지 오브젝트를 전송함으로써, 상기 데이터를 목적지 파이프라인(74_1~74_n) 모두로 동시에 상기 데이터를 전송한다. 메시지 오브젝트로부터 데이터 및 소프트-구성 명령을 복구하는 것은 앞서 언급한 발명의 명칭이 "PIPELINE ACCELERATOR FOR IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD" 인 미국 특허출원 제10/683,929호에 더 설명되어 있다.

다른 실시예에서, 처리-유닛이 프로세싱 또는 파이프라인(74)이 다수인 경우, 각각의 파이프라인은 라우터(61)로부터 메시지 오브젝트를 수신한다. 비록 라우터(61)가 타겟 파이프라인(74)으로만 메시지 오브젝트를 이상적으로 전송해야 하지만, 상기 타겟 파이프라인은 계속 헤더를 분석하고 그것이 상기 메시지의 의도된 수신측인지 아닌지를 결정한다. 그러한 결정으로 잠재적인 메시지 라우팅 에러, 예를 들어, 에러를 식별한다. 만일 메시지 오브젝트가 타겟 파이프라인(74)을 위한 것이라면, 그 파이프라인은 메시지를 판독하고 복구된 데이터/명령을 처리한다. 그러나, 만일 메시지 오브젝트가 타겟 파이프라인(74)을 위한 것이 아니라면, 그 파이프라인은 상기 메시지 오브젝트를 위한 처리를 무시하고 에러 라우팅이 생성했음을 나타내는 새로운 메시지를 호스트 프로세서(42)로 발행한다. 라우팅 에러의 처리에 대해서는 앞서 언급한 발명의 명칭이 "COMPUTING MACHINE HAVING IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD" 인 미국 특허출원 제10/684,053호에 설명되어 있다.

다음으로, 파이프라인 가속기(44)는 상기 메시지 오브젝트로부터 복구되어 입력 데이터 및/또는 명령을 처리한다.

데이터를 위해, 상기 목적지 파이프라인 또는 파이프라인들(74)은 상기 데이터상의 연산 또는 연산들을 수행한다. 도 2를 참고로 설명한 바와 같이, 파이프라인(74)은 프로그램 명령을 실행하지 않으므로, 파이프라인은 상기 파이프라인 클록의 주파수와 거의 동일한 레이트로 상기 데이터를 처리할 수 있다.

제1 실시예에서, 하나의 파이프라인(74)이 상기 입력 데이터를 처리함으로써 결과 데이터를 생성한다.

제2 실시예에서, 다수의 파이프라인(74)이 상기 입력 데이터를 직렬로 처리함으로써 결과 데이터를 생성한다. 예를 들어, 파이프라인(74)은 상기 입력 데이터에서 제1 연산을 수행하여 제1 중간 데이터를 생성한다. 다음으로, 파이프라인(74_2)이 상기 제1 중간 데이터상에서 제2 연산을 수행하여 제2 중간 데이터를 생성하고, 이렇게 순서대로 그 체인 내의 최종 파이프라인(74)이 상기 결과 데이터를 생성할 때 까지 진행된다.

제3 실시예에서, 다수의 파이프라인(74)은 상기 입력 데이터를 병렬로 처리함으로써 상기 결과 데이터를 생성한다. 예를 들어, 파이프라인(74_1)은 상기 입력 데이터의 첫번째 세트에서 제1 연산을 수행함으로써 결과 데이터의 제1 세트를 생성한다. 같이 시간에서, 파이프라인(74_2)은 상기 입력 데이터의 두번째 세트에서 제2 연산을 수행함으로써 결과 데이터의 제2 세트를 생성한다. 이와 같은 방식으로 진행된다.

선택적으로, 파이프라인(74)은 상기 세 개의 실시예의 어느 조합에 따라 상기 입력 데이터로부터 결과 데이터를 생성한다. 예를 들어, 파이프라인(74_1)은 상기 입력 데이터의 첫번째 세트에서 제1 연산을 수행함으로써 결과 데이터의 제1 세트를 생성한다. 동시에, 파이프라인(74_2 및 74_n)은 상기 입력 데이터의 두번째 세트에서 제2 및 제3 연산을 직렬로 수행함으로써 결과 데이터의 제2 세트를 생성한다.

상기 실시예 및 선택안 중 어느 것에서, 하나의 파이프라인(74)은 다중 연산을 수행한다. 예를 들어, 파이프라인(74_1)은 데이터를 수신하고, 상기 수신된 데이터에서 제1 연산을 수행함으로써 제1 중간 데이터를 생성하고, 상기 제1 중간 데이터를 임시 저장하고, 상기 제1 중간 데이터에서 제2 연산을 수행함으로써 제2 중간 데이터를 생성하고, 이후 최종 데이터가 생성될 때 까지 진행된다. 파이프라인(74_1)이 상기 제1 연산을 수행하는 것부터 상기 제2 연산, 등등을 수행하는 것 까지 스위치를 하게 하는 다양한 기술들이 있다. 그러한 기술은 앞서 언급한 발명의 명칭이 "PIPELINE ACCELERATOR FOR IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD" 인 미국 특허출원 제10/683,929호(대리인 관리번호 1934-13-3)에 설명되어 있다.

소프트-구성 명령을 위해서, 가속기(44)는 메시지 헤더에 의한 표시된(도시하지 않음)대응하는 소프트-구성 레지스터내의 비트를 설정한다. 상기 설명한 바와같이, 일반적으로 이들 비트를 설정하는 것은 이 비트에 그것의 상호접속 레이아웃을 변화시키지 않고 가속기(44)의 기능을 변화시킨다. 이것은 어부 핀을 입력핀 또는 출력핀으로 설정하거나 또는 어드레싱 모드를 설정하기 위해 프로세서의 제어 레지스터 내의 비트를 설정하는 것과 유사하다. 또한, 가속기(44)에 의해 수행된 더 다른 소프트-구성 명령 또는 연산은 데이터를 소프트-구성된 레지스터 또는 메모리로 데이터를 로드한다. 가속기(44)의 소프트-구성은 앞서 언급한 발명의 명칭이 "PIPELINE ACCELERATOR FOR IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD" 인 미국 특허출원 제10/683,929호에 더 설명되어 있다.

다음에, 파이프라인 가속기(44)는 그 이후의 처리를 위해 라우터(61)(또는 라우터가 생략되면)를 경유하여 호스트 프로세서(42)에 결과 데이터를 제공한다.

선택적으로, 가속기(44)는 직접적으로 출력 포트(60)를 경유하거나 또는 간접적으로 라우터(61)(만일 있는 경우), 버스(50), 호스트 프로세서(42) 및 출력 포트(58)를 경유하여 원격 목적지(도 5)에게 그 결과 데이터를 제공한다. 따라서, 이 선택적인 실시예에서, 가속기(44)에의해 생성된 결과 데이터가 최종 처리된 데이터이다.

가속기(44)가 상기 결과 데이터를 호스트 프로세서(42)로 제공하면 - 그 이상의 처리로 또는 원격 장치(도 5)를 통과시키는 -, 메시지 처리기(64)에 의해 생성된 메시지 오브젝트와 동일한 포맷을 갖는 메시지 오브젝트 내에 이 데이터를 전송한다. 메시지 처리기(64)에 의해 생성된 메시지 오브젝트와 같이, 가속기(44)에 의해 생성된 메시지 오브젝트들에는 결과 데이터의 목적지 및 우선순위를 특정하는 헤더들을 포함한다. 예를 들면, 헤더는 포트(58)를 경유하여 원격 장치에 결과 데이터를 통과시키기 위하여 메시지 처리기(64)에게 명령하기도 하거나, 또는, 처리 유닛(62)에 의해 실행된 프로그램의 어떤 부분이 데이터의 제어를 처리하도록 특정하기도 한다. 동일한 메시지 포맷을 사용함으로써, 가속기(44)는 호스트 프로세서(42)와 동일한 인터페이스 계층을 가지고 있다. 이것은 피어-벡터 머신을 디자인하고 수정하는 것을 용이하게 해 주고, 특히, 만약 인터페이스 계층이 업계 표준일 경우 그러하다.

파이프라인 가속기(44) 및 파이프라인(66)의 구조와 동작은 앞서 언급한 발명의 명칭이 "PIPELINE ACCELERATOR FOR IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD" 인 미국 특허출원 제10/683,929호에 더 설명되어 있다.

호스트 프로세서를 가지고 파이프라인 가속기로부터의 수신 및 처리

가속기(44)로부터 메시지 오브젝트를 수신하는 경우, 메시지 처리기(64)는 우선 메시지 헤더를 판독하고 목적된 데이터를 표시된 목적지로 안내한다.

만약, 상기 헤더가 상기 데이터가 포트(58)를 통해 원격 장치(도 5)로 통과할 것임을 나타낸다면, 메시지 처리기(64)는 그 데이터를 포트(58)로 직접 제공하거나 또는 인터페이스 메모리(48) 또는 상기 버퍼로부터 상기 포트(58)까지 다른 메모리 내에 형성된 포트 FIFO 버퍼(도시하지 않음)로 제공한다. 다중 포트(58) 및 다수의 각각 원격 장치들도 예상된다.

그러나, 만약 헤더가 상기 처리 유닛(62)이 상기 데이터를 더 처리해야 함을 나타낸다면, 메시지 처리기(62)는 이 데이터를 상기 데이터의 처리를 제어하는 처리-유닛 프로그램의 일부에 대응하는 인터페이스 메모리(48)의 위치내에 저장한다. 보다 특별하게는, 동일한 헤더가 상기 처리 유닛(54)에 의해 실행된 프로그램의 일부가 상기 데이터의 처리를 제어하는지를 표시한다. 따라서, 메시지 처리기(64)는 상기 데이터를 이 프로그램 부분에 대응하는 인터페이스 메모리(48)의 위치(FIFO 같은)에 저장한다.

상기 설명한 바와 같이, 인터페이스 메모리(48)는 상기 가속기(44)과 처리 유닛(42) 사이의 버퍼로서 작동하고, 그래서 처리 유닛이 가속기와 동기화되지 않으면 데이터 전송을 한다. 예를 들어, 이러한 동기화가 되지 않는 것은 가속기(44)가 처리 유닛(62) 보다 데이터 처리를 빨리하는 경우 생성한다. 인터페이스 메모리(48)를 사용함으로써, 가속기(44)는 처리 유닛(62)의 늦은 응답에 의해 늦춰지지 않는다. 이것은 또한 처리 유닛의 랜덤한 인터럽트로의 응답 시간이 부정확한 것과 관련된 유효하지 않은 패널티(penalty)를 막아준다. 상기 가속기(44)의 처리 유닛(62)에 의한 부정확한 처리는 출력 메시지가 설계자에게 a) 백업 출력 메시지를 위한 저장 및 처리, 또는 b) 백업 메시지가 겹쳐 쓰여지는 것을 막기 위한 파이프라인 전체의 아이들링(idling) 제어 중 어느 하나를 강요하여 가속기의 설계를 불필요하게 복잡하게 할 것이다. 따라서, 가속기(44)과 처리 유닛(62) 사이의 버퍼로서 동작하는 인터페이스 메모리(48)를 사용하는 것은 a) 가속기의 설계를 쉽게하고, b) 가속기에 기반시설 요구를 줄여주고 보다 큰 PLIC 애플리케이션로 유지할 수 있고, c) 가속기가, 출력 데이터가 더 느린 프로세서에 의해 "막혀지(blocked)" 않기 때문에 빠른 구동으로 능률적이 되는 다양한 장점을 가지게 된다.

그래서, 메시지 처리기(64)가 인터페이스 메모리(48)내에 저장하는 데이터를 위해, 처리 유닛(62)은 상기 인터페이스 메모리로부터 데이터를 검색한다. 상기 처리 유닛(62)은 상기 인터페이스 메모리(48)를 폴(poll)하여 새로운 데이터가 특정 위치에 도달한 경우 또는 메시지 처리기(64)가 데이터의 도달의 처리 유닛을 인식하는 인터럽트 또는 다른 신호를 생성하는 때를 결정한다. 한 실시예에서, 처리 유닛(62)이 데이터를 검색하기 전에, 메시지 처리기(64)는 상기 데이터를 포함하는 메시지 오브젝트를 생성한다. 좀 더 특별하게, 설계자는 처리 유닛(62)에 의해 실행된 프로그램을 설계하여 메시지 오브젝트 내에 데이터를 수신하게 한다. 따라서, 메시지 처리기(64)는 데이터만을 저장하는 대신 인터페이스 메모리(48)내에 메시지 오브젝트를 저장할 수 있다. 그러나, 메시지 오브젝트는 보통 데이터가 포함되어 있는 것 보다 상당히 많은 메모리를 점유하고 있다. 따라서, 메모리를 절약하기 위해서는, 메시지 처리기(64)는 파이프라인 가속기(44)로부터 메시지 오브젝트를 판독하고, 메모리(48) 내에 상기 데이터를 저장하고, 그 다음, 처리 유닛(62)이 데이터를 수신할 준비가 되면 메시지 오브젝트를 새로 생성한다. 그리고 나서, 처리 유닛(62)은 메시지 오브젝트를 판독하고 상기 데이터를 메시지 헤더내에 식별된 프로그램 부분의 제어하에 처리한다.

다음으로, 처리 유닛(62)은 프로그램의 목적지 부분의 제어하에 상기 검색된 데이터를 처리하고, 처리된 데이터를 생성하고, 이 처리된 데이터를, 상기 처리된 데이터의 의도된 목적지에 대응하는 인터페이스 메모리(48)의 위치에 저장한다.

이어서, 메시지 처리기(64)는 상기 처리된 데이터를 검색하고 이를 표기된 목적지로 제공한다. 처리된 데이터를 검색하기 위해서, 메시지 처리기(64)는 메모리(48)를 폴하여 상기 데이터가 언제 도달했는지, 또는 처리 유닛(62)이 인터럽트 또는 다른 신호로서 상기 데이터의 도달을 메시지 처리기에에 통보했는지를 결정한다. 상기 처리된 데이터를 의도된 목적지로 제공하기 위해서, 메시지 처리기(64)는 상기 데이터를 포함하는 메시지 오브젝트를 생성하고, 이 메시지 오브젝트를 상기 데이터의 더 이상의 처리를 위해 가속기(44)로 되돌려 보낸다. 또는, 상기 처리기(64)는, 처리 유닛(62)에 의한 더 이상의 처리를 위해 상기 데이터를 포트(68) 또는 상기 메모리(48)의 다른 위치로 전송한다.

가속기(44)로부터 데이터의 상기 호스트 프로세서의 수신 및 처리는 앞서 언급한 발명의 명칭이 "COMPUTING MACHINE HAVING IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD" 인 미국 특허출원 제10/684,053호에 더 설명되어 있다.

피어-백터 머신을 이용한 더 다른 처리 기술

도 3을 계속 참고하면, 호스트 프로세서(44)가 데이터를 수신하고 처리한 다음 이 데이터를 더 다른 처리를 위해 파이프라인 가속기(44)로 전송하는 상기 설명한 실시예와 다른 예가 있다.

한 대안에서는, 호스트 프로세서(44)가 상기 데이터의 적어도 일부의 처리를 모두 수행하고는 이 데이터를 더 다른 처리를 위해 파이프라인 가속기(44)로 전송하지 않는다.

더 다른 대안에서는, 파이프라인 가속기(44)가 포트(50)를 통해 원격 장치(도 5)로부터 직접 원 데이터를 수신하고 상기 원 데이터를 처리한다. 가속기(44)는 상기 처리된 데이터를 포트(60)를 통해 상기 원격 장치로 직접 되돌려 보내거나 또는 더 다른 처리를 위해 상기 호스트 프로세서(44)로 전송한다. 후자의 경우, 가속기(44)는 상기 설명한 바와 같이 메시지 오브젝트에 상기 데이터를 캡슐화(encapsulate)한다.

더 다른 대안에서는, 가속기(44)에, 하드와이어드 파이프라인(74)에 더해서, 디지털 신호 처리기(DSP)와 같은 하나 또는 그 이상의 명령-실행 프로세서를 포함시켜 파이프라인의 수-프로그램 가능성을 수행한다.

피어-백터 머신의 예시적 수행

도 3을 계속 참고하면, 한 실시예에서, 파이프라인 버스(50)는 표준 133MHz PCI 버스이고, 파이프라인(74)에는 하나 또는 그 이상의 표준 PMC 카드가 포함되어 있으며, 메모리(52)는 각각이 개개의 PMC 카드상에 위치하는 플래쉬 메모리이다.

피어-백터 머신의 예시적 애플리케이션

도 4는 본 발명의 일 실시예에 따른 피어-백터 머신(40)을 통합하는 소나 시스템(80)의 블록 다이어그램이다. 머신(40)에 더하여, 상기 시스템(80)에는 소나 신호를 송수신하는 변환 소자(transducer element)(84_1-84_n), 디지털-아날로그 변환기(DAC)(86_1-86_n), 아날로그-디지털 변환기(ADC)(88_1-88_n) 및 데이터 인터페이스(90)이 포함되어 있다. 소나 신호를 생성하고 처리하는 것은 종종 수학적으로 집중되는 기능이기 때문에, 머신(40)은 이러한 기능들을 종래의 컴퓨팅 머신 - 도 1의 다중-프로세서 머신(10)과 같은 - 이 도 3을 참고로 상기 설명한 바와 같이 주어진 클록 주파수에서 할 수 있는 것보다 빠르고 유효하게 수행할 수 있다.

연산의 전송 모드 동안, 어레이(82)는 소나 신호를 물과 같은 매체(도시하지 않음)로 전송한다. 먼저, 피어-백터 머신(40)은 포트(92)상에서 수신된 원 신호 데이터를 상기 어레이 소자(84) 각각의 하나를 위한 n 디지털 신호로 변환한다. 이들 신호의 크기와 위상은 어레이(82)의 전송-빔 패턴을 나타낸다. 다음으로, 머신(40)은 이들 디지털 신호를, 이들 신호를 각각의 DAC(80)에 제공하여 각각의 아날로그 신호로 변환하는 인터페이스(90)로 제공한다. 예를 들어, 인터페이스(90)는 머신(40)으로부터 상기 디지털 신호를 직렬로 수신하고, 이들의 n 개 모두를 수신하고 버퍼할 때 까지 이들 신호를 저장하고, 이들 순차적인 신호를 각각의 DAC(80)으로 동시에 제공하는 버퍼 역할을 한다. 그리고 나서, 변환 소자(84)가 이들 아날로그 신호를 소나 신호의 빔을 형성하기 위해 상호 간섭하는 각각의 음파로 변환한다.

연산의 수신 모드 동안에, 어레이(82)는 상기 매체(도시하지 않음)로부터 소나 신호를 수신한다. 수신된 소나 신호는 원격 오브젝트에 의해 반사된 상기 전송된 소나 신호의 일부 및 환경 및 상기 원격 오브젝트에 의해 방출된 사운드 에너지로 구성되어 있다. 우선, 변환 소자(84)가 상기 소나 신호를 구성하는 각각의 음파를 수신하고, 이들 음파를 n 개의 아날로그 신호로 변환하고, 이들 아날로그 신호를 n 개의 각각의 디지털 신호로 변환을 위해 ADC(88)로 제공한다. 다음으로, 인터페이스(90)가 이들 디지털 신호를 피어-백터 머신(40)으로 제공하여 처리되게 한다. 예를 들어, 인터페이스(90)는 ADC(88)로부터 상기 디지털 신호를 병렬로 수신한 다음 이들 신호를 머신(40)으로 직렬로 제공하는 버퍼로서 동작한다. 상기 디지털 신호에서 상기 머신(40)이 수행하는 처리는 어레이(82)의 수신-빔 패턴을 나타낸다. 필터링, 핸드 시프팅, 스펙트럼 변환(푸리에 변환 등), 및 폰볼루션과 같은 추가적인 처리 단계가 상기 디지털 신호에 적용된다. 그리고 나서 머신(40)은 포트(94)를 통해 상기 처리된 데이터를 디스플레이 장치와 같은 다른 장치로 제공하여 위치한 오브젝트를 볼 수 있게 한다.

소나 시스템(80)을 참고로 설명하였으나, 소나 시스템 이외의 시스템도 피어-백터 머신(40)에 통합된다.

앞의 설명으로부터 당업자는 본 발명을 실시하거나 활용할 수 있다. 본 발명의 실시예들을 당업자는 다양하게 변형시킬 수 있다는 것은 분명하고, 본 발명의 요지와 범위를 벗어나지 않고 다른 실시예 및 응용 분야에 적용할 수 있다. 따라서, 본 발명의 실시예들은 본 발명을 제한하기 위한 것이 아니고 여기에서 공개한 원리와 특징에 넓게 해석되야 할 것이다.

(57) 청구의 범위

청구항 1.

프로그램을 실행하고, 상기 프로그램에 응답하여 제1 호스트 데이터를 생성하도록 동작 가능한 호스트 프로세서; 및

상기 호스트 프로세서와 결합되고, 상기 제1 호스트 데이터를 수신하고 상기 제1 호스트 데이터로부터 제1 파이프라인 데이터를 생성하도록 동작 가능한 파이프라인 가속기를 구비하는 것을 특징으로 하는 퍼어-벡터 머신.

청구항 2.

청구항 1에 있어서,

상기 호스트 프로세서는,

제2 데이터를 수신하고,

상기 제2 데이터로부터 제1 호스트 데이터를 생성하도록 동작 가능한 것을 특징으로 하는 퍼어-벡터 머신.

청구항 3.

청구항 1에 있어서,

상기 호스트 프로세서는,

상기 파이프라인 가속기로부터 상기 제1 파이프라인 데이터를 수신하고,

상기 제1 파이프라인 데이터를 처리하도록 동작 가능한 것을 특징으로 하는 퍼어-벡터 머신.

청구항 4.

청구항 1에 있어서,

상기 호스트 프로세서는,

상기 제1 파이프라인 가속기로부터 상기 제1 파이프라인 데이터를 수신하고,

상기 제1 파이프라인 데이터로부터 상기 제1 호스트 데이터를 생성하도록 동작 가능한 것을 특징으로 하는 퍼어-벡터 머신.

청구항 5.

청구항 1에 있어서,

상기 호스트 프로세서 및 상기 파이프라인 가속기와 결합되어 있고 제1 메모리부를 가지는 인터페이스 메모리를 더 구비하고,

상기 호스트 프로세서는,

상기 제1 메모리부내에 상기 제1 호스트 데이터를 저장하고,

상기 제1 호스트 데이터를 상기 제1 메모리부로부터 상기 파이프라인 가속기로 제공하도록 동작 가능한 것을 특징으로 하는 피어-벡터 머신.

청구항 6.

청구항 1에 있어서,

상기 호스트 가속기 및 상기 파이프라인 가속기과 결합되어 있고 제1 및 제2 메모리부를 가지는 인터페이스 메모리를 더 구비하고,

상기 호스트 프로세서는,

상기 제1 메모리부 내에 상기 제1 호스트 데이터를 저장하고,

상기 제1 호스트 메모리를 상기 제1 메모리부로부터 상기 파이프라인 가속기로 제공하고,

상기 파이프라인 가속기로부터 상기 제1 파이프라인 데이터를 수신하고,

상기 제2 메모리부 내에 상기 제1 파이프라인 데이터를 저장하고,

상기 제2 메모리부로부터 상기 호스트 프로세서까지 상기 제1 파이프라인 데이터를 검색하고, 그리고

상기 제1 파이프라인 데이터를 처리하도록 동작 가능한 것을 특징으로 하는 피어-벡터 머신.

청구항 7.

청구항 1에 있어서,

상기 호스트 프로세서는 상기 파이프라인 가속기를 구성하도록 동작 가능한 것을 특징으로 하는 피어-벡터 머신.

청구항 8.

청구항 1에 있어서,

상기 파이프라인 가속기는 프로그램가능한-로직 집적회로를 포함하는 것을 특징으로 하는 피어-벡터 머신.

청구항 9.

제1 파이프라인 데이터를 생성하도록 동작 가능한 파이프라인 가속기; 및

상기 파이프라인 가속기와 결합되고 프로그램을 실행하도록 동작 가능하며, 상기 프로그램에 응답하여 상기 제1 파이프라인 데이터를 수신하고 상기 제1 파이프라인 데이터로부터 제1 호스트 데이터를 생성하도록 동작 가능한 것을 특징으로 하는 피어-벡터 머신.

청구항 10.

청구항 9에 있어서,

상기 파이프라인 가속기는,

제2 데이터를 수신하고,

상기 제2 데이터로부터 상기 제1 파이프라인 데이터를 생성하도록 동작 가능한 것을 특징으로 하는 피어-벡터 머신.

청구항 11.

청구항 9에 있어서,

상기 파이프라인 가속기는,

상기 호스트 프로세서로부터 상기 제1 호스트 데이터를 수신하고,

상기 제1 호스트 데이터를 처리하도록 동작 가능한 것을 특징으로 하는 피어-벡터 머신.

청구항 12.

청구항 9에 있어서,

상기 파이프라인 가속기는,

상기 호스트 프로세서로부터 상기 제1 호스트 데이터를 수신하고,

상기 제1 호스트 데이터로부터 상기 제1 파이프라인 데이터를 생성하도록 동작 가능한 것을 특징으로 하는 피어-벡터 머신.

청구항 13.

청구항 9에 있어서,

상기 파이프라인 가속기 및 상기 호스트 프로세서와 결합되고 제1 메모리부를 갖는 인터페이스 메모리를 더 구비하고,

상기 호스트 프로세서는,

상기 파이프라인 가속기로부터의 상기 제1 파이프라인 데이터를 상기 제1 메모리부 내에 저장하고,

상기 제1 메모리부로부터 상기 제1 파이프라인 데이터를 검색하도록 동작 가능한 것을 특징으로 하는 피어-벡터 머신.

청구항 14.

청구항 9에 있어서,

상기 파이프라인 가속기 및 상기 호스트 프로세서와 결합되고 제1 및 제2 메모리부를 가지는 인터페이스 메모리를 더 구비하고,

상기 호스트 프로세서는,

상기 파이프라인 가속기로부터의 상기 제1 파이프라인 데이터를 상기 제1 메모리부 내로 저장하고,

상기 제1 부로부터 상기 제1 파이프라인 데이터를 검색하고,

상기 제2 메모리부 내에 상기 제1 호스트 데이터를 저장하고,

상기 제2 메모리부로부터의 상기 제1 호스트 데이터를 상기 파이프라인 가속기에 제공하도록 동작하능하며,

상기 파이프라인 가속기는 상기 제2 메모리부로부터 수신된 상기 제1 호스트 데이터를 처리하도록 동작 가능한 것을 특징으로 하는 피어-벡터 머신.

청구항 15.

청구항 9에 있어서,

상기 호스트 프로세서는 상기 파이프라인 가속기를 구성하도록 동작 가능한 것을 특징으로 하는 피어-벡터 머신.

청구항 16.

원 데이터를 생성하도록 동작 가능한 장치;

상기 장치와 결합되어 있고 프로그램을 실행할 수 있으며, 상기 프로그램에 응답하여 상기 원 데이터로부터 호스트 데이터를 생성하도록 동작 가능한 호스트 프로세서; 및

상기 호스트 프로세서와 결합되어 있고 상기 호스트 데이터를 수신하도록 동작 가능하며 상기 호스트 데이터로부터 파이프라인 데이터를 생성하도록 동작 가능한 파이프라인 가속기를 구비하는 것을 특징으로 하는 시스템.

청구항 17.

원 데이터를 생성하도록 동작 가능한 장치;

상기 장치와 결합되어 있고 상기 원 데이터로부터 파이프라인 데이터를 생성하도록 동작 가능한 파이프라인 가속기; 및

상기 파이프라인 가속기와 결합되어 있고 프로그램을 실행하도록 동작 가능하며, 상기 프로그램에 응답하여, 상기 파이프라인 데이터를 수신하고 상기 파이프라인 데이터로부터 호스트 데이터를 생성하도록 동작 가능한 호스트 프로세서를 구비하는 것을 특징으로 하는 시스템.

청구항 18.

호스트 프로세서를 가지고 프로그램을 실행하여 제1 호스트 데이터를 생성하는 단계; 및

파이프라인 가속기를 가지고 상기 제1 호스트 데이터로부터 제1 파이프라인 데이터를 생성하는 단계를 구비하는 것을 특징으로 하는 방법.

청구항 19.

청구항 18에 있어서,

원 데이터를 수신하는 단계를 더 구비하고,

상기 제1 호스트 데이터를 생성하는 단계는 상기 원 데이터로부터 상기 제1 호스트 데이터를 생성하는 단계를 포함하는 것을 특징으로 하는 방법.

청구항 20.

청구항 18에 있어서,

상기 제1 호스트 데이터를 생성하는 단계는 상기 제1 파이프라인 데이터로부터 상기 제1 호스트 데이터를 발생하는 단계를 포함하는 것을 특징으로 하는 방법.

청구항 21.

청구항 18에 있어서,

상기 호스트 프로세서를 가지고 상기 프로그램을 실행하여 상기 제1 파이프라인 데이터로부터 제2 호스트 데이터를 생성시키는 단계를 더 구비하는 것을 특징으로 하는 방법.

청구항 22.

청구항 18에 있어서,

상기 호스트 프로세서를 가지고 상기 프로그램을 실행하여 상기 파이프라인 가속기를 구성하는 단계를 더 구비하는 것을 특징으로 하는 방법.

청구항 23.

파이프라인 가속기를 가지고 제1 파이프라인 데이터를 생성하는 단계; 및

호스트 프로세서를 가지고 프로그램을 실행하여 상기 제1 파이프라인 데이터로부터 제1 호스트 데이터를 생성하는 단계를 구비하는 것을 특징으로 하는 방법.

청구항 24.

청구항 23에 있어서,

원 데이터를 수신하는 단계를 더 구비하고,

상기 제1 파이프라인 데이터를 생성하는 단계는 상기 원 데이터로부터 상기 제1 파이프라인 데이터를 생성하는 단계를 포함하는 것을 특징으로 하는 방법.

청구항 25.

청구항 23에 있어서,

상기 제1 파이프라인 데이터를 생성하는 단계는 상기 제1 호스트 데이터로부터 상기 제1 파이프라인 데이터를 생성하는 단계를 포함하는 것을 특징으로 하는 방법.

청구항 26.

청구항 23에 있어서,

상기 파이프라인 가속기를 가지고 상기 제1 호스트 데이터로부터 제2 파이프라인 데이터를 생성하는 단계를 더 구비하는 것을 특징으로 하는 방법.

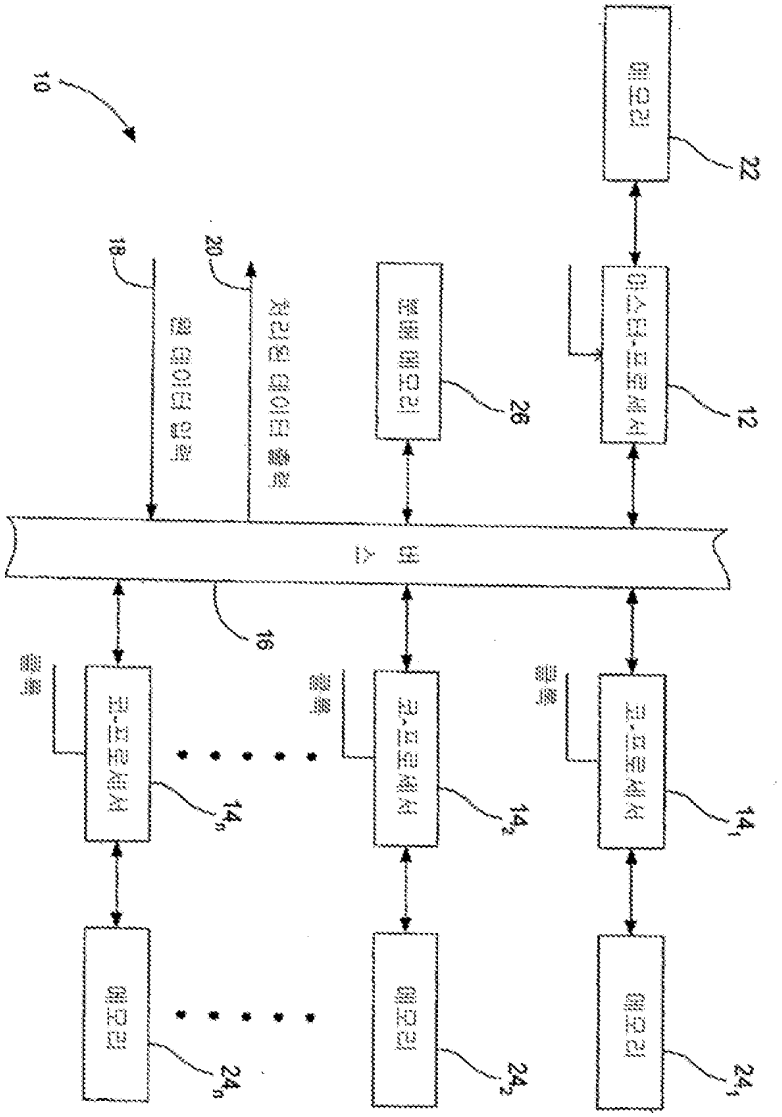
청구항 27.

청구항 23에 있어서,

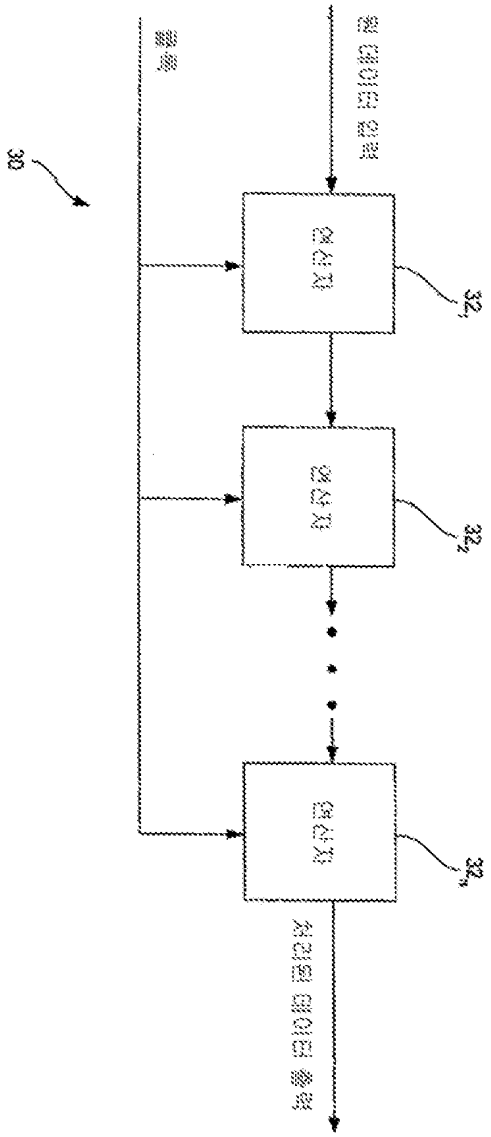
상기 호스트 프로세서를 가지고 상기 프로그램을 실행하여 상기 파이프라인 가속기를 구성하는 단계를 더 구비하는 것을 특징으로 하는 방법.

도면

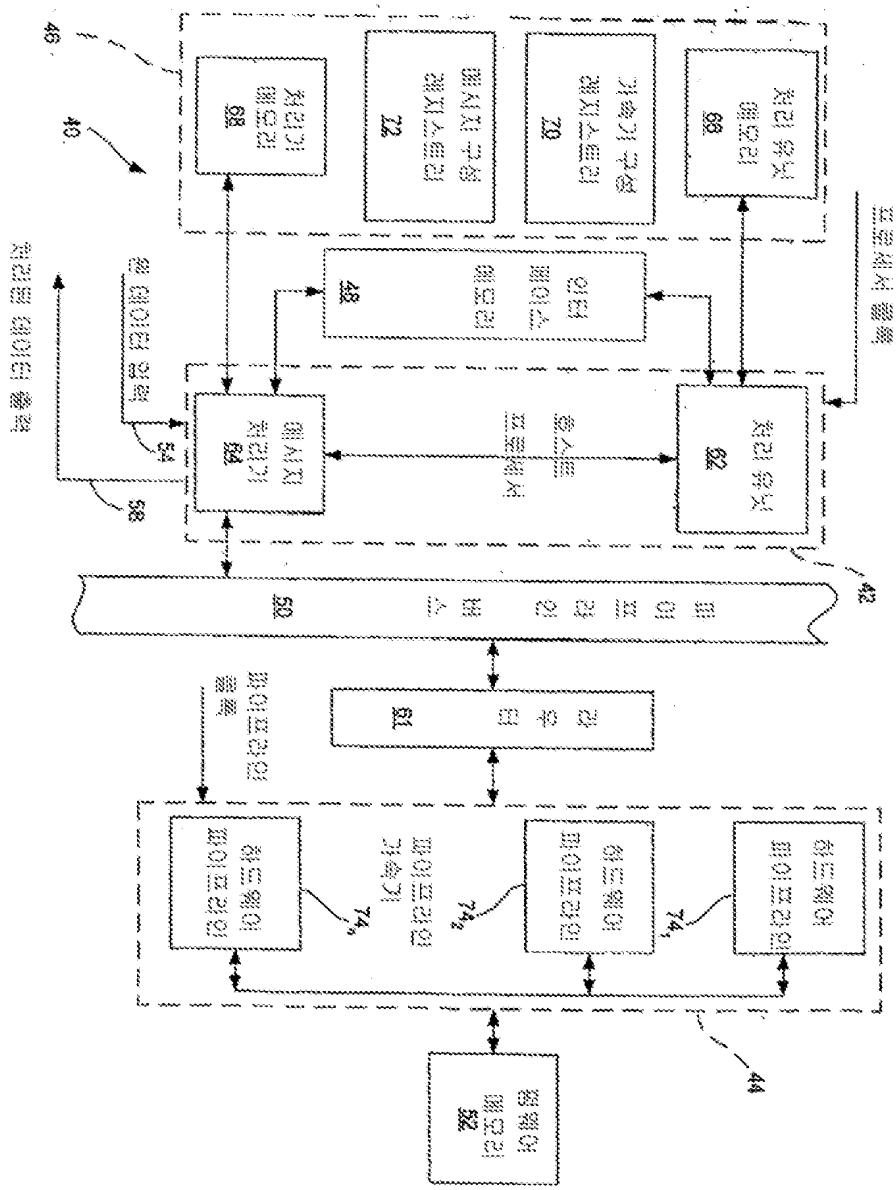
도면1


10
22
메모리
12
마스터-프로세서
26
분배 메모리
18
입력 데이터 입력
20
처리된 데이터 출력
버스
16
14₁
코-프로세서
14₂
코-프로세서
14ₙ
코-프로세서
24₁
메모리
24₂
메모리
24ₙ
메모리

도면2



도면3


46
40
처리기 메모리 88
레지스터 구성 메모리 22
기술기 구성 레지스터 20
처리 유닛 메모리 66
프로세서 클록
인터페이스 메모리 48
처리된 데이터 출력
출력 데이터 입력
54
58
메시지 처리기 64
송수신 프로세서
처리 유닛 62
42
50
버스 인터페이스 유닛
81
파이프라인 클록
하드웨어 파이프라인 74n
파이프라인 기술기
하드웨어 파이프라인 74a
하드웨어 파이프라인 74a
44
확장 메모리 82

도면4